Listing of Claims

1. (Currently Amended) A method for enabling COBOL programs for distributed processing, comprising:

providing a COBOL technical layer stored on a computer-readable medium for use by [[a]] COBOL programs, the COBOL technical layer enabling a distributed processing module through a call made to a function of an operating system, wherein the COBOL technical layer defines a bit level mapping of the call to interface with the operating system;

~~providing~~ executing a COBOL program stored on a computer-readable medium; and

employing, by the executing COBOL program, the distributed processing module ~~to enable~~ such that the executing COBOL program [[to]] performs distributed processing, the COBOL program and the COBOL technical layer operating in the same runtime environment.

2. (Canceled)

3. (Currently Amended) The method of Claim 1, wherein the COBOL technical layer is further defined as a COBOL complier enabling a plurality of distributed processing modules.

4. (Currently Amended) The method of Claim 1, wherein the COBOL technical layer is further defined as a COBOL pre-complier enabling a plurality of distributed processing modules.

5. (Currently Amended) The method of Claim 1, wherein the method further comprises providing a mainframe computer system wherein the mainframe computer system executes the COBOL program and provides the COBOL technical layer~~ are provided~~.

6. (Currently Amended) The method of Claim 1, wherein the method further comprises:

   providing a library of callable routines in the COBOL technical layer, wherein the distributed processing module is one of the callable routines; and

   linking the library of the COBOL technical layer to the COBOL program.

7. (Currently Amended) The method of Claim 6, wherein providing the library includes:

   generating an object code using the COBOL technical layer; and

   linking the object code of the COBOL technical layer to the COBOL program.

8. (Currently Amended) The method of Claim 6, wherein a source code of the COBOL technical layer is written in COBOL language.

9. (Currently Amended) The method of Claim 6, wherein a source code of the COBOL technical layer is written in Assembly language.

10. (Currently Amended) The method of Claim 1, wherein the COBOL technical layer is further defined as including a plurality of distributed processing modules.

11. (Currently Amended) The method of Claim 10, wherein the plurality of distributed processing modules of the COBOL technical layer ~~are further~~ include at least a socket module.

12. (Currently Amended) The method of Claim 11, wherein the COBOL program and the COBOL technical layer operate in the runtime environment defined as at least the distributed processing module providing at least some commonly used functions and variables for use by the COBOL program while the COBOL program is ~~operating~~ executing.

13. (Original) The method of Claim 1, wherein providing the COBOL program further comprises:

writing, in COBOL language, the COBOL program to perform a distributed processing task.

14. (Currently Amended) The method of Claim 1, wherein the method further comprises performing, by the COBOL program using the distributed processing module of the COBOL technical layer, the distributed processing task.

15. (Original) The method of Claim 1, wherein the distributed processing module is further defined as a COBOL program subsystem.

16. (Original) The method of Claim 1, wherein the distributed processing module is further defined as a COBOL program library routine call.

17. (Original) The method of Claim 1, wherein the distributed processing module is further defined as a COBOL program compiler enabled function.

18. (Currently Amended) A system for enabling distributed and asynchronous processing by COBOL programs on a computer, comprising:

    ~~a computer system;~~

    a COBOL extension layer stored on a computer-readable medium enabling at least one module operable for a distributed and asynchronous processing task through a call made to a function of an operating system, wherein the COBOL extension layer defines a bit level mapping of the call to interface with the operating system; and

    a program stored on a computer-readable medium written in COBOL programming language~~ for the computer system~~, the program employing the module to perform the distributed and asynchronous processing task ~~on the computer system~~; and

    a computer system that executes the COBOL program and performs the distributed and asynchronous processing task in accordance with the employment of the module by the COBOL program, wherein the COBOL extension layer and the COBOL program operate in the same runtime environment.

19. (Original) The system of Claim 18, wherein the COBOL extension layer is further defined as enabled by a compiler.

20. (Original) The system of Claim 18, wherein the COBOL extension layer is further defined as a library linked to the COBOL program, the library having at least one routine callable from the COBOL program.

21. (Original) The system of Claim 18, wherein the computer system is a mainframe.

22. (Original) The system of Claim 18, wherein the distributed and asynchronous processing task is further defined as at least one of a socket task, a shared memory task, a thread task and a task, a signal handler task, an events task, a semaphore task, and a mutex task.

23. (Currently Amended) A COBOL compiler stored on a computer-readable medium that compiles COBOL programs such that the ~~for enabling~~ COBOL programs [[to]] perform distributed and asynchronous processing tasks upon execution, the COBOL compiler comprising:

~~an engine to compile a COBOL program; and~~

a distributed and asynchronous processing module to enable distributed and asynchronous processing by the COBOL programs through a call made to a function of an operating system, wherein the distributed and asynchronous processing module defines a bit level mapping of the call to interface with the operating system; and

an engine that compiles the COBOL programs using the distributed and asynchronous processing module,

wherein the COBOL compiler and the COBOL programs operate in the same runtime environment.

24. (Original) The COBOL compiler of Claim 23, wherein the distributed and asynchronous processing modules of the compiler includes a module to enable the COBOL program for socket communications.

25. (Original) The COBOL compiler of Claim 23, wherein the distributed and asynchronous processing modules of the compiler includes a module to enable the COBOL program for pipe communications.

26. (Original) The COBOL compiler of Claim 23, wherein the distributed and

asynchronous processing modules of the compiler includes a module to enable the COBOL program for sharing memory between COBOL programs.

27. (Original) The COBOL compiler of Claim 23, wherein the distributed and asynchronous processing modules of the compiler includes a module to enable the COBOL program for processing threads.

28. (Original) The COBOL compiler of Claim 23, wherein the distributed and asynchronous processing modules of the compiler includes a module to enable the COBOL program to use a queue.

29. (Original) The COBOL compiler of Claim 28, wherein the queue is further defined as a message queue.

30. (Original) The COBOL compiler of Claim 28, wherein the queue is further defined as a memory queue.

31. (Original) The COBOL compiler of Claim 23, wherein the distributed and asynchronous processing modules of the compiler includes a module to enable the COBOL program for operating system signal handling.

32. (Original) The COBOL compiler of Claim 23, wherein the distributed and asynchronous processing modules of the compiler includes a module to enable the COBOL program for managing events.

33. (Original) The COBOL compiler of Claim 23, wherein the distributed and asynchronous processing modules of the compiler includes a module to enable the COBOL program to use semaphores.

34. (Original) The COBOL compiler of Claim 23, wherein the distributed and asynchronous processing modules of the compiler includes a module to enable the COBOL program to use mutexes.

35. (Currently Amended) A method for enabling COBOL programs for asynchronous processing, comprising:

providing a COBOL technical layer stored on a computer-readable medium for use by [[a]] COBOL programs, the technical layer enabling an asynchronous processing module through a call made to a function of an operating system, wherein the COBOL technical layer defines a bit level mapping of the call to interface with the operating system;

~~providing~~ executing a COBOL program stored on a computer-readable medium; and

employing, by the executing COBOL program, the asynchronous processing module ~~to enable~~ such that the executing COBOL program [[to]] performs asynchronous processing, the COBOL program and the COBOL technical layer operating in the same runtime environment.

36. (Canceled)

37. (Original) The method of Claim 35, wherein the technical layer is further defined as a complier enabling the asynchronous processing module.

38. (Original) The method of Claim 35, wherein the technical layer is further defined as a pre-complier enabling the asynchronous processing module.

39. (Original) The method of Claim 35, wherein the method further comprises providing a mainframe computer system wherein the COBOL program and technical layer operate in the runtime environment.